Status Page Application install guide

Welcome to your final workshop! This is the code file, that contain the explanation and instructions how to install your application. Good luck!

Requirements:

OS – Linux Centos or Ubuntu with **Python3.10 installed**

DBs:

- PostregeSql 10 or higher
- Redis 4 or higher

Application components:

- Status-Page – Django application
- Gunicorn
- HTTP Server – Nginx or Apache

Networking requirements:

- Server running Status page should have ports 8000 and 443 open for user connections.
- Server running PostgreSql should be open in database port to the server running the application (default is 5432)
- Server running Redis should be open in Redis port to server running the application (default 6379)

Prerequisites:

PostgreSQL server

Setup PostgreSQL in any way you like (server install, docker or managed service).

Server should:

- Use PostgreSQL version 10 or higher
- A database called statuspage
- A user called statuspage
- The user should have complete privileges on the database.

You can run from inside the database (From inside Postgres, not the server) to configure:

CREATE DATABASE statuspage;

CREATE USER statuspage WITH PASSWORD '<enter here a secure password>';

GRANT ALL PRIVILEGES ON DATABASE statuspage TO statuspage;

Redis Server

Setup Redis in any way you like (server install, docker or managed service).

Use Redis version 4.0 or higher.

Status-Page Installation

Install System Packages

Begin by installing all system packages required by Status-Page and its dependencies.

Python 3.10 or later required.

Status-Page requires Python 3.10 or later.

Centos
```
sudo yum install -y gcc libxml2-devel libxslt-devel libffi-devel
libpq-devel openssl-devel redhat-rpm-config
```

Ubuntu
```
sudo apt install -y python3 python3-pip python3-venv python3-dev
build-essential libxml2-dev libxslt1-dev libffi-dev libpq-dev libssl-
dev zlib1g-dev
```

Before continuing, check that your installed Python version is at least 3.10:

```
python3 -V
```

Upload Status-Page

Create the base directory for the installation. We'll use /opt/status-page for now.

```
sudo mkdir -p /opt/status-page/
cd /opt/status-page/
```

Upload the content of status page application code directory to the directory created above.

Create the Status-Page System User

Create a system user account named status-page. We'll configure the WSGI and HTTP services to run under this account.

Centos
```
sudo groupadd --system status-page
sudo adduser --system -g status-page status-page
```

Ubuntu
```
sudo adduser --system --group status-page
```

Configuration

Move into the Status-Page configuration directory and make a copy of `configuration_example.py` named `configuration.py`. This file will hold all of your local configuration parameters.

```
cd /opt/status-page/statuspage/statuspage/
sudo cp configuration_example.py configuration.py
```

Open `configuration.py` with your preferred editor to begin configuring Status-Page. Status-Page offers many configuration parameters, but only the following four are required for new installations:

- `ALLOWED_HOSTS`
- `DATABASE`
- `REDIS`
- `SECRET_KEY`

ALLOWED_HOSTS

This is a list of the valid hostnames and IP addresses by which this server can be reached. You must specify at least one name or IP address. (Note that this does not restrict the locations from which Status-Page may be accessed: It is merely for HTTP host header validation.)

```
ALLOWED_HOSTS = ['status-page.example.com', '192.0.2.123']
```

If you are not yet sure what the domain name and/or IP address of the Status-Page installation will be, you can set this to a wildcard (asterisk) to allow all host values:

```
ALLOWED_HOSTS = ['*']
```

DATABASE

This parameter holds the database configuration details. You must define the username and password used when you configured PostgreSQL. If the service is running on a remote host, update the HOST and PORT parameters accordingly. See the configuration documentation for more detail on individual parameters.

```
DATABASE = {
    'NAME': 'status-page',          # Database name
    'USER': 'status-page',          # PostgreSQL username
    'PASSWORD': 'abcdefgh123456',   # PostgreSQL password
    'HOST': 'localhost',            # Database server
    'PORT': '',                     # Database port (leave blank for
default)
    'CONN_MAX_AGE': 300,            # Max database connection age
(seconds)
}
```

REDIS

Redis is a in-memory key-value store used by Status-Page for caching and background task queuing. Redis typically requires minimal configuration; the values below should suffice for most installations. See the configuration documentation for more detail on individual parameters.

Note that Status-Page requires the specification of two separate Redis databases: tasks and caching. These may both be provided by the same Redis service, however each should have a unique numeric database ID.

```
REDIS = {
    'tasks': {
        'HOST': 'localhost',      # Redis server
        'PORT': 6379,             # Redis port
        'PASSWORD': '',           # Redis password (optional)
        'DATABASE': 0,            # Database ID
        'SSL': False,             # Use SSL (optional)
    },
    'caching': {
        'HOST': 'localhost',
        'PORT': 6379,
        'PASSWORD': '',
        'DATABASE': 1,            # Unique ID for second database
        'SSL': False,
    }
}
```

SECRET_KEY

This parameter must be assigned a randomly-generated key employed as a salt for hashing and related cryptographic functions. (Note, however, that it is *never* directly used in the encryption of secret data.) This key must be unique to this installation and is recommended to be at least 50 characters long. It should not be shared outside the local system.

A simple Python script named generate_secret_key.py is provided in the parent directory to assist in generating a suitable key:

```
python3 ../generate_secret_key.py
```

SECRET_KEY values must match

In the case of a highly available installation with multiple web servers, SECRET_KEY must be identical among all servers in order to maintain a persistent user session state.

When you have finished modifying the configuration, remember to save the file.

Optional Requirements

All Python packages required by Status-Page are listed in requirements.txt and will be installed automatically. Status-Page also supports some optional packages. If desired, these packages must be listed in local_requirements.txt within the Status-Page root directory.

Run the Installation and Upgrade Script

Once Status-Page has been configured, we're ready to proceed with the actual installation. We'll run the packaged upgrade script (`upgrade.sh`) to perform the following actions:

- Create a Python virtual environment
- Installs all required Python packages
- Run database schema migrations
- Builds the documentation locally (for offline use)
- Aggregate static resource files on disk

```
sudo /opt/status-page/upgrade.sh
```

Note that **Python 3.10 or later is required** for Status-Page v2.0 and later releases. If the default Python installation on your server is set to a lesser version, pass the path to the supported installation as an environment variable named `PYTHON`. (Note that the environment variable must be passed *after* the `sudo` command.)

```
sudo PYTHON=/usr/bin/python3.10 /opt/status-page/upgrade.sh
```

Note

Upon completion, the upgrade script may warn that no existing virtual environment was detected. As this is a new installation, this warning can be safely ignored.

Create a Super User

Status-Page does not come with any predefined user accounts. You'll need to create a super user (administrative account) to be able to log into Status-Page. First, enter the Python virtual environment created by the upgrade script:

```
source /opt/status-page/venv/bin/activate
```

Once the virtual environment has been activated, you should notice the string (venv) prepended to your console prompt.

Next, we'll create a superuser account using the `createsuperuser` Django management command (via `manage.py`). Specifying an email address for the user is not required, but be sure to use a very strong password.

```
cd /opt/status-page/statuspage
python3 manage.py createsuperuser
```

Test the Application

At this point, we should be able to run Status-Page's development server for testing. We can check by starting a development instance:

```
python3 manage.py runserver 0.0.0.0:8000 --insecure
```

If successful, you should see output similar to the following:

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 30, 2021 - 18:02:23
Django version 3.2.6, using settings 'statuspage.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Next, connect to the name or IP of the server (as defined in ALLOWED_HOSTS) on port 8000; for example, http://127.0.0.1:8000/dashboard/. You should be greeted with the Status-Page login page. Try logging in using the username and password specified when creating a superuser.

Note
By default RHEL based distros will likely block your testing attempts with firewalld. The development server port can be opened with `firewall-cmd` (add `--permanent` if you want the rule to survive server restarts):

```
firewall-cmd --zone=public --add-port=8000/tcp
```

Not for production use

The development server is for development and testing purposes only. It is neither performant nor secure enough for production use. **Do not use it in production.**

Warning
If the test service does not run, or you cannot reach the Status-Page home

page, something has gone wrong. Do not proceed with the rest of this guide until the installation has been corrected.

Type `Ctrl+c` to stop the development server.

Getting a production ready server:

Gunicorn

Like most Django applications, Status-Page runs as a WSGI application behind an HTTP server. This documentation shows how to install and configure gunicorn (which is automatically installed with Status-Page) for this role, however other WSGI servers are available and should work similarly well.

Configuration

Status-Page ships with a default configuration file for gunicorn. To use it, copy /opt/status-page/contrib/gunicorn.py to /opt/status-page/gunicorn.py. (We make a copy of this file rather than pointing to it directly to ensure that any local changes to it do not get overwritten by a future upgrade.)

```
sudo cp /opt/status-page/contrib/gunicorn.py /opt/status-page/gunicorn.py
```

While the provided configuration should suffice for most initial installations, you may wish to edit this file to change the bound IP address and/or port number, or to make performance-related adjustments. See the Gunicorn documentation for the available configuration parameters.

systemd Setup

We'll use systemd to control gunicorn, Status-Page's background worker scheduler and Status-Page's background worker process. First, copy `contrib/status-page.service`, `contrib/status-page-scheduler.service` and `contrib/status-page-rq.service` to the `/etc/systemd/system/` directory and reload the systemd daemon:

```
sudo cp -v /opt/status-page/contrib/*.service /etc/systemd/system/
sudo systemctl daemon-reload
```

Then, start the `status-page`, `status-page-scheduler` and `status-page-rq` services and enable them to initiate at boot time:

```
sudo systemctl start status-page status-page-scheduler status-page-rq
sudo systemctl enable status-page status-page-scheduler status-page-rq
```

You can use the command `systemctl status status-page` to verify that the WSGI service is running:

```
systemctl status status-page.service
```

You should see output similar to the following:

```
● status-page.service - Status-Page WSGI Service
     Loaded: loaded (/etc/systemd/system/status-page.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-10-30 17:54:22 UTC; 14h ago
       Docs: https://docs.status-page.dev/
```

```
   Main PID: 1573 (gunicorn)
      Tasks: 19 (limit: 4683)
     Memory: 666.2M
     CGroup: /system.slice/status-page.service
             ├─1573 /opt/status-page/venv/bin/python3 /opt/status-
page/venv/bin/gunicorn >
             ├─1579 /opt/status-page/venv/bin/python3 /opt/status-
page/venv/bin/gunicorn >
             ├─1584 /opt/status-page/venv/bin/python3 /opt/status-
page/venv/bin/gunicorn >
...
```

Note

If the Status-Page service fails to start, issue the command journalctl -eu status-page to check for log messages that may indicate the problem.

Once you've verified that the WSGI workers are up and running, move on to HTTP server setup.

HTTP Server Setup

This documentation provides example configurations for both nginx and Apache, though any HTTP server which supports WSGI should be compatible.

Info

For the sake of brevity, only Ubuntu 20.04 instructions are provided here. These tasks are not unique to Status-Page and should carry over to other distributions with minimal changes. Please consult your distribution's documentation for assistance if needed.

Obtain an SSL Certificate

To enable HTTPS access to Status-Page, you'll need a valid SSL certificate. You can purchase one from a trusted commercial provider or obtain one for free from Let's Encrypt. Both the public certificate and private key files need to be installed on your Status-Page server in a location that is readable by the `status-page` user.

The command below can be used to generate a self-signed certificate for testing purposes, however it is strongly recommended to use a certificate from a trusted authority in production. Two files will be created: the public certificate (`status-page.crt`) and the private key (`status-page.key`). The certificate is published to the world, whereas the private key must be kept secret at all times.

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
```

```
-keyout /etc/ssl/private/status-page.key \
-out /etc/ssl/certs/status-page.crt
```

The above command will prompt you for additional details of the certificate; all of these are optional.

HTTP Server Installation

Option A: nginx

Begin by installing nginx:

```
sudo apt install -y nginx
```

Once nginx is installed, copy the nginx configuration file provided by Status-Page to `/etc/nginx/sites-available/status-page.conf`. Be sure to replace `status-page.example.com` with the domain name or IP address of your installation. (This should match the value configured for `ALLOWED_HOSTS` in `configuration.py`.)

```
sudo cp /opt/status-page/contrib/nginx.conf /etc/nginx/sites-available/status-page.conf
```

Then, delete `/etc/nginx/sites-enabled/default` and create a symlink in the `sites-enabled` directory to the configuration file you just created.

```
sudo rm /etc/nginx/sites-enabled/default
sudo ln -s /etc/nginx/sites-available/status-page.conf /etc/nginx/sites-enabled/status-page.conf
```

Finally, restart the nginx service to use the new configuration.

```
sudo systemctl restart nginx
```

Option B: Apache

Begin by installing Apache:

```
sudo apt install -y apache2
```

Next, copy the default configuration file to /etc/apache2/sites-available/. Be sure to modify the `ServerName` parameter appropriately.

```
sudo cp /opt/status-page/contrib/apache.conf /etc/apache2/sites-available/status-page.conf
```

Finally, ensure that the required Apache modules are enabled, enable the status-page site, and reload Apache:

```
sudo a2enmod ssl proxy proxy_http headers
sudo a2ensite status-page
sudo systemctl restart apache2
```

Confirm Connectivity

At this point, you should be able to connect to the HTTPS service at the server name or IP address you provided.

Info

Please keep in mind that the configurations provided here are bare minimums required to get Status-Page up and running. You may want to make adjustments to better suit your production environment.

Warning

Certain components of Status-Page (such as the display of rack elevation diagrams) rely on the use of embedded objects. Ensure that your HTTP server configuration does not override the `X-Frame-Options` response header set by Status-Page.

Troubleshooting

If you are unable to connect to the HTTP server, check that:

- Nginx/Apache is running and configured to listen on the correct port.
- Access is not being blocked by a firewall somewhere along the path. (Try connecting locally from the server itself.)

If you are able to connect but receive a 502 (bad gateway) error, check the following:

- The WSGI worker processes (gunicorn) are running (`systemctl status status-page` should show a status of "active (running)")
- Nginx/Apache is configured to connect to the port on which gunicorn is listening (default is 8001).
- SELinux is not preventing the reverse proxy connection. You may need to allow HTTP network connections with the command `setsebool -P httpd_can_network_connect 1`